A Nonparametric Outlier Detection for Effectively Discovering Top-N Outliers from Engineering Data

Hongqin Fan[1], Osmar R. Zaïane[2], Andrew Foss[2], and Junfeng Wu[2]

[1] Department of Civil Engineering, University of Alberta, Canada
[2] Department of Computing Science, University of Alberta, Canada

Abstract. We present a novel resolution-based outlier notion and a nonparametric outlier-mining algorithm, which can efficiently identify top listed outliers from a wide variety of datasets. The algorithm generates reasonable outlier results by taking both local and global features of a dataset into consideration. Experiments are conducted using both synthetic datasets and a real life construction equipment dataset from a large building contractor. Comparison with the current outlier mining algorithms indicates that the proposed algorithm is more effective.

1 Introduction

The term "outlier" can refer to any single data point of dubious origin or disproportionate influence. *Given a set of observations X, an outlier is an observation that is an element of this set but which is inconsistent with the majority of the data or inconsistent with a sub-group of X to which the element is meant to be similar*. The above definition has two implications: outlier vis-à-vis the majority; and outlier vis-à-vis a group of neighbours. Whether it is an interesting contaminant or dubious data entry, an outlier is often considered noise, which can have a harmful effect on statistical analysis.

Attempts have been made to remove the noisy data using various outlier mining approaches; in one example, Raz et al. [1] designed an expert system to automatically detect unlikely vehicles and erroneously classified ones from weigh-in-motion data. In contrast to noisy data, some relevant outliers contain important information on system malfunction, mismanagement, or even unpredictable phenomena (environmental or geological disaster), which should be detected for further investigation rather than discarded. In either of these two cases, the inconsistent records should first be identified as much as possible from the dataset. Data validation (range validation, single variate pattern validation etc.) can only filter out a small portion of outliers. Traditional statistical approaches including multivariate outlier detection are not applicable due to their pre-assumption of certain statistical distributions, which may not exist for datasets containing multiple clusters.

Outlier mining techniques in data mining seem to be viable solutions for outlier detection in engineering applications. Some of the popular algorithms include among others a distance-based outlier mining algorithm by Knorr and Ng [2]; a local outlier mining algorithm by Breunig et al. [3] and a connectivity-based mining algorithm by

W.K. Ng, M. Kitsuregawa, and J. Li (Eds.): PAKDD 2006, LNAI 3918, pp. 557 – 566, 2006.

Tang et al. [4]. However these algorithms are not widely accepted in civil engineering disciplines because they do not cater to the special features of engineering datasets, including:

- Current outlier mining algorithms need domain-dependent parameters, but these parameters are not known a priori;
- Current outlier mining algorithms need some parameters, which can only be obtained and tuned through tremendous trial-and-error effort. This is not practical for frequent time-changing applications and thus cannot be an integrated part of a real-time decision support system.
- Current outlier mining algorithms are capable of mining either global or local outliers while the engineering dataset usually contains loosely bounded clusters. It is difficult in this case to differentiate local from global outliers.
- In engineering applications, there exists a need for ranking the top-listed outliers. This is where our major focus is.

In this paper, we present a Resolution-Based outlier (RB-outlier) notion and an associated outlier detection algorithm efficient for engineering applications. The RB-outlier notion is proposed based on a nonparametric clustering algorithm called TURN* by Foss and Zaïane with the same idea of resolution change [5]. The proposed algorithm can detect and rank top-N outliers from any kind of dataset without the need for input parameters.

We also compare RB-outlier with DB-outlier and Local density based outlier (LOF-outlier) mining algorithms using both synthetic datasets and a construction equipment dataset from a large building contractor. Our experimental results show that the RB-outlier mining algorithm generates equivalent or better results than the other two competitive algorithms on all the datasets while benefiting from the absence of input parameters; the RB-outlier results seem to combine the results from both DB-outlier which looks for global outliers and LOF-outlier which looks for local outliers. Analysis on the detected outliers from the equipment datasets shows that these combined results make more sense for engineering datasets.

2 Related Work

Hawkins defines an outlier as "*an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism*" [6]. Traditionally outlier detection in engineering disciplines depends on statistical approaches. After fitting a data series into a bell-shaped statistical distribution, those data points located far away from the mean (e.g. 3 standard deviations) are deemed outliers; multivariate outlier detection techniques can help identify outliers within a multivariate dataset. Other commonly used techniques are quartile methods, and visualization methods using scatter plot, etc. With regard to the data complexity and sheer data volume in engineering systems, outlier detection using statistical approaches is very inefficient and even impractical due to their limitations such as the difficulties with handling higher dimensional data, and the necessary assumption of distributions.

A distance-based definition of outliers was first proposed by Knorr and Ng. They introduced DB-outlier to identify outliers from a large database (i.e. with high dimensions and high data volume) [2]. A DB-outlier is defined as follows: "*An object O in a dataset T is considered a DB(p,D)-outlier if at least a fraction p of the objects in T lies greater than distance D from O*". The authors claim that this definition generalizes the notion of outlier defined in statistical tests for standard distributions. DB-outlier tends to find outliers in global context, which means some important outliers deviated from their local clusters are probably missed if a large number of isolated points or loosely packed clusters appear. While Knorr and Ng's definition is distribution free, it lacks a mechanism to rank outliers. Without violating the original notion of DB(p,D) outlier, Ramaswamy et al. further propose to rank each point based on its distance to its k^{th} nearest neighbour, and use a partition-based algorithm to efficiently mine top-N outliers from a large database [7]. For them, *the outliers are the top n data elements whose distance to the k^{th} nearest neighbour is greatest*. This definition also eliminates the need to estimate an appropriate distance D.

Another popular algorithm is a local density based outlier-mining algorithm proposed by Breunig et al. [3]. A Local Outlier Factor (LOF) is assigned for each object with respect to its surrounding neighbourhood. The LOF value depends on how the data points are closely packed in its local reachable neighbourhood. These points deep inside a dense cluster have a LOF value of approximately 1 while the isolated points have a much higher value. The authors claim that this definition also catches the spirit of the outlier definition given by Hawkins [6]. The local outlier notion seems more reasonable than DB-outlier because each data point can be measured with a numerical factor based on how the data is deviated from its genuine cluster. Therefore the outliers can be ranked as per their LOF values.

Tang et al. improved to some extent the LOF definition by using their Connectivity-based Outlier Factor (COF) for a dataset containing low density patterns. In such a case, LOF would not be effective to measure the density of an outlier with respect to its sparse neighbourhood [4].

The biggest hurdle of effectively applying these afore-mentioned outlier-mining algorithms in the engineering domain is the determination of their input parameters. All the parameters should be either known *a priori* or estimated and optimized by trial-and-error with the help of expert opinions. In particular, LOF is very sensitive to its parameter MinPts and DB-outlier results vary greatly when D and p change. Subjective results make it difficult to implement these outlier-mining algorithms for outlier detection in engineering applications.

3 Resolution-Based Outlier

TURN* is a nonparametric clustering algorithm [5]. The optimum clustering of a dataset can be obtained automatically based on resolution change: when the resolution changes on a dataset, the clusters in the dataset redistribute. All the objects are in the same cluster when the resolution is very low, meanwhile every object is a single cluster when the resolution is very high, and therefore the optimum clustering can be achieved at a point between these two extreme scenarios. The "TURN-CUT" technique was introduced to detect this critical point during the resolution change by looking for a plateau in the curve of the differential of some collected cluster statistics [5].

The same observation holds when viewing all the objects in the dataset from the perspective of outliers. All the objects are outliers when the resolution is high enough to warrant no neighbours (by distance measure) for any objects in the dataset; meanwhile all the objects are "inliers" when the resolution is low enough to have all the objects close-packed in a single cluster. If the resolution of a dataset changes, different outliers demonstrate different clustering-related behaviors during resolution change; those objects more isolated, with less neighbours, and far away from large data communities are more liable to be outliers. On the other hand, the top outliers will be merged into a cluster later when the resolution is decreased. As a result, the accumulated cluster-related properties collected on one object can be used to measure its degree of outlyingness relative to its close neighbourhood and community (reachable neighbourhoods). We first define the neighbourhood of an object:

Definition 1. Neighbourhood of Object O:

> *If an Object O has a nearest neighbouring points P along each dimension in k-dimensional dataset D and the distance between P and O is less or equal to 1, then P is defined as the close neighbour of O, all the close neighbours of P are also classified as the close neighbours of O, and so on. All these connected objects are classified as the same neighbourhood.*

The threshold value is taken as 1 to measure whether two points are close enough to become neighbours. The absolute value of this threshold is not important because the pair-wise distances between points are relative measurements during resolution change. The algorithm finds the maximum resolution S_{max} at which all the points are far enough from each other to be non-neighbours, and the minimum resolution S_{min} at which all the points are close enough to be neighbours.

Secondly we define the resolution-based outlier factor for each object:

Definition 2. Resolution-Based Outlier Factor (ROF):

> *If the resolution of a dataset changes consecutively between maximum resolution where all the points are non-neighbours, and minimum resolution where all the points are neighbours, the resolution-based outlier factor of an object is defined as the accumulated ratios of sizes of clusters containing this object in two consecutive resolutions.*

$$ROF(O) = \sum_{i=1}^{R} \frac{ClusterSize(O, r_{i-1}) - 1}{ClusterSize(O, r_i)}$$

Where r_0, $r_2 \ldots r_{i \ldots} r_R$ are the resolutions at each step, R is the total number of resolution change steps from S_{max} to S_{min}, CluserSize(O,r) is the number of objects in the cluster containing object O at a resolution r.

At each resolution, we cluster the points based on the distance between every two objects and the neighbourhood definition. The time complexity of clustering at each resolution is O(NLogN) as demonstrated in [5]. The cluster size of each object is set to 1 at the beginning (i.e. at Max. resolution), then the cluster size increases for the object whenever the object gets merged, or the cluster containing the object merges other objects at the next lower resolution.

The cluster size at the previous resolution is reduced by 1 for ease of comparison, which will set the ROF of an object to 0 before the object gets merged. Top N outliers are the N elements with the lowest ROF value.

Universally defining an outlier is somewhat controversial and is often subjective, relative to the application at hand. While considering locality and globality, our definition still embodies the essence of the accepted definition given by Hawkins [6] with regard to deviation from other observations.

4 Resolution-Based Outlier Mining Algorithm

Using the definitions of close neighbourhood and ROF in the previous section, a resolution-based outlier detection and ranking algorithm (RB-MINE) is proposed for mining top-N outliers in a dataset with multiple numerical attributes.

The resolution change ratio r is a percentile value used for computing the resolution change step size. The reason why this parameter is not considered an input parameter for the nonparametric clustering algorithm TURN* is explained in [5]. The same argument can be applied here to the RB-outlier mining algorithm. Indeed our experimental tests show the resolution change ratio, so long as it is kept in a moderate range, has a minor effect on the outlier results. Thus the user does not need to spend much effort to fine-tune this parameter, unlike the current popular outlier mining algorithms. Moreover, this ratio need not be static. It can decrease dynamically starting with a large step declining to a smaller step progressively. We found in our extensive experiments that fixing this resolution change to a static 10% gave good results for a wide range of data.

RB-CLUSTER

Given a resolution *r* and a dataset D:

1. Scale the coordinates using current resolution *r*.
   Current Coordinates = Original Coordinates * *r*
2. For each object O
   For the objects within a threshold distance of 1 from O, find the closest neighbours in each direction (+,-) along every dimension. (this can be done with a sort on each dimension)
3. Pick an unlabeled object and give it a new cluster label *C*.
   Initialize its neighbourhood chain n*Chain* and set the cluster size of *C* to 1.
4. Scan this object's close neighbours. For each neighbour:
   If the neighbour is unlabeled, give the neighbour the same cluster label *C*, add the neighbour to n*Chain*, and increase the cluster size of *C* by 1.
   If the neighbour has already been labeled as *C'* (*C'* ≠ *C*), change the label of all points in cluster *C'* to *C*. Increase the cluster size of *C* by the number of objects contained in cluster *C'*, then delete records on cluster *C'*.
5. Move to the next object on *nChain*. Repeat 4 until all objects on *nChain* are checked.
6. Record the size of cluster *C*.
7. Repeat (3)~(6) until all objects are labeled.
8. For each object *p*, update the ROF value.

Fig. 1. RB-CLUSTER component algorithm

RB-MINE

Given a dataset D and the specified number N of top outliers

1. Find the maximum resolution, S_{max}, at which no close neighbours can be found for each object, and the minimum resolution, S_{min}, at which all the points are close neighbours in the same neighbourhood.
2. Starting at S_{max}, initialize the ROF value as 0 for each object.
3. Update $r_i = r_{i-1} + (S_{max} - S_{min})*\Delta r$, ($\Delta r$ is the resolution changing ratio).
4. Run RB-CLUSTER to cluster the objects at resolution r_i.
5. Update the ROF value for each object.
6. Rank objects in an increasing order of ROF, obtain top N outliers.

Fig. 2. RB-MINE algorithm

The finalized RB-outlier algorithm is comprised of a component algorithm RB-CLUSTER in Figure 1, and an overall algorithm RB-MINE in Figure 2. RB-CLUSTER clusters objects in the dataset at each resolution step while RB-MINE steers the mining process through the change of resolution and collection of ROF for each object.

5 Comparison of RB-Outlier with DB-Outlier and LOF-Outlier

One of the primary factors determining the outlier mining results is the outlier notion, which describes what an outlier is and assigns it a measuring factor, if it is possible.

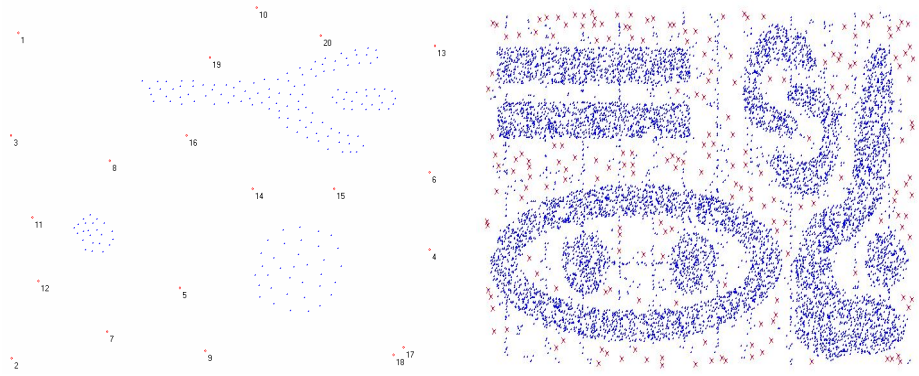
For examples, DB(D,p) outlier evaluates the inconsistency of an object by judging if there are sufficient number of objects within D (or the distance to its k^{th} nearest neighbour is closer than D, where k is the (1-p) percent of the total number of objects), therefore the algorithm searches for the specified number of nearest points in a global context; LOF-outlier measures the degree of outlyingness by taking only a restricted neighbourhood into account, LOF varies depending on how an object is deviated from its "best-guess" cluster in a local context.

The RB-outlier notion measures how an object deviates from its close neighbourhood. The definition of neighbourhood implies that this neighbourhood actually includes a series of chained neighbourhoods (we call it a "community"), as such, RB-outlier measures an object against its degree of outlyingness by taking both "global" and "local" features into account.

6 Experimental Results

To validate the RB-outlier mining results and compare with those detected by DB-outlier and LOF-outlier mining algorithms, we implemented all three algorithms in the same C++ development environment, and conducted experiments on a number of synthetic datasets as well as a real world construction equipment dataset obtained from a large building contractor. This section summarizes our experimental results and comparative analysis on a token 200-tuple 2D synthetic dataset, a 10,000-tuple 2D synthetic dataset, and a 1033-tuple 3D construction equipment dataset.

A Synthetic 2D dataset of four clusters with a total number of 200 objects: The token dataset shown in Figure 3.A, contains four distinct clusters and 20 outliers. This dataset, used in many clustering research projects for validating clustering results can be equally applied for visual validation of outlier detection. We run the three algorithms separately on this dataset, each identifying the top-10 and top-20 outliers and marking up these points in the density plot. DB-outlier was optimal with p=97/200 and LOF-outlier with MinPts=3. Figure 3.A summarizes the top outliers detected by RB-outliers.



A: Top-20 outliers identified by RB-outlier in a 200-tuple synthetic dataset

B: Top-200 outliers identified by RB-outlier in a 10,000-tuple synthetic dataset

Fig. 3. Top outliers identified by RB-outlier in two synthetic datasets

Among the top-10 outliers, eight out of the top-10 outliers are the same with the three algorithms; visual judgment on the density plot confirms the "outlyingness" of the identified objects and the eight objects voted unanimously as outliers are truly isolated as compared with others. This finding indicates that all the three algorithms can effectively find outliers from this dataset. However there are some differences between the three sets of results:

1. Though top-3 outliers are exactly the same in the three sets of results, the other five unanimously identified outliers have different rankings in their perspective results. The five outliers have higher rankings in RB-outlier list.
2. The outliers No. 4 and 5 in LOF-outlier results are interpreted differently by DB-outlier and RB-outlier. In the DB-outlier set, No. 4 is ranked as No. 10 and No.5 is not included in the top-10; nevertheless neither of them is included in RB-outlier results at the top-10 level. This observation indicates that LOF-outlier tends to take objects in a small isolated cluster as outliers, the same tendency can be observed with DB-outlier. This finding can be explained by the definition for the minimum number of neighbouring points in LOF-outlier and DB-outlier: a small cluster containing number of objects less than this specified parameter tends to be classified as "a cluster of outliers."

The afore-mentioned conclusions are further enhanced by increasing the number of outliers in the top list. For example, if we look at the top-20 outliers in Figure 3.A, they are indeed detected by all the algorithms. However, RB-outlier shifts the unanimously identified outliers to higher rankings in its top-20 list.

Precision and recall are two measures in information retrieval that could be used as criteria for evaluating the outlier results. Precision is defined as the percentage of correct outliers in the set of detected outliers (i.e. the total number of correctly detected outliers divided by the number of labeled outliers); recall is the percentage of all known outliers correctly detected (i.e. total number of correctly detected outliers divided by the total number of existing outliers). For top-20, Precision and Recall are the same and all three algorithms achieve 100%. However, scrutinizing at a lower n for top-outliers, RB-outlier always has better or equivalent precision and recall.

A synthetic 10,000 object dataset containing nine clusters of different shapes: This dataset is also widely used for visually validating clustering results. The dataset contains 9 clusters of difference shapes and significant noise. This experiment aims to verify the capability of RB-outlier to distinguish outliers from "inliers" in a relatively large set and in the presence of clusters of arbitrary shapes.

In this experiment, we compared the top-200 outlier results identified by LOF-outlier and RB-outlier respectively, as marked up in Figure 3.B. Though both identify similar objects as outliers in this dataset, a detailed observation reveals that LOF-outlier biases objects in locally sparse areas, and RB-outlier takes local sparsity and global sparsity into consideration when picking up outliers. The LOF algorithm assigns a higher outlier factor for the object whose local density is lower than its neighbours. Therefore, when the number of outliers to detect is increased, LOF-outlier marks the objects at the edge of dense clusters as outliers, while RB-outlier tends to pick up outliers from isolated objects. RB-outlier's performance is indeed the sought for behaviour in construction engineering data.

A 3D construction equipment management dataset: To test-drive the RB-outlier mining algorithm and validate its usefulness in engineering datasets, we conducted experiments on a three-dimensional construction equipment dataset obtained from a large building contractor. The dataset includes characteristic attributes for 1033 pieces of equipment in the contractor's equipment fleet. Three numerical attributes are defined for each unit in addition to the identify attribute: the yearly repair/maintenance cost (yearly cost), the rate of charge, and the age. The objective of the experiments is to identify top listed inconsistent units from the dataset. The inconsistency of these units indicates the abnormal combination of the three attributes for a unit with respect to its similar equipment sub-group;

Using DB-outlier, LOF-outlier and RB-outlier algorithms, we identify the top-20 outliers from the selected equipment fleet. A comparison of the three sets of outliers determines that 11 out of 20 are unanimously identified as outliers by the three algorithms in their top-20 lists. LOF-outlier generates very similar results to DB-outlier with 16 being the same out of the top-20. This is not surprising because a large number of isolated objects appear in this dataset. LOF definition becomes similar to the DB-outlier notion for a sparsely distributed dataset: both start by looking for minimum number of neighbouring objects. Two additional units are identified as outliers by both LOF-outlier and RB-outlier; but no addition outlying units are identified by

both DB-outlier and RB-outlier besides the 11 common units. RB-outlier moves all the 11 common units to higher rankings and adds 6 new units in the top-20 outlier list. Analysis of the 6 added units against their individual equipment sub-group and the entire fleet confirmed their interestingness.

The inability of identifying some outstanding outliers by DB-outlier and LOF-outlier mining algorithms can be illustrated from their outlier notions: these two algorithms evaluate the degree of outlying by drawing a hyper-sphere around each object; and the number of objects inside the hyper-sphere influences the outlier measurement of the object. The outliers harbored inside the concave of a cluster can not be identified efficiently based on this rational, subsequently some outliers become missing in the results if the two algorithms are applied on a real life dataset which may contain clusters of any arbitrary shape.

Among the top-20 outliers, Unit# 505-401, a soil cement plant (300 to 600 TPH), is detected as No.1 outlier by DB-outlier and No. 2 outlier by LOF-outlier, but is not identified as an outlier by RB-outlier. Detailed analysis of this unit finds out it is the only unit in this equipment class, therefore it is not a "true" outlier in the application. Other outliers in top-20 of DB-outliers (no. 16 and 17) are also the only units in their respective equipment class; No. 18 of LOF-outliers is identified as outlier because it is in a small 2-object cluster. These outlying units in DB-outlier and LOF-outlier results are of no particular interests; their appearance somehow degrades the general quality of the top-N outlier mining results. On the contrary, all the 11 common outlying units, the three found by RB-outlier and by another method, and the 6 additional units unique to the RB-outlier results are indeed inconsistent units relative to their individual equipment sub-group. If we consider the 11 common units as "true" outliers, and look at the top-20 or less in the outlier results, the recall is generally better for RB-outlier when we compare the three algorithms.

Execution time is not reported or discuss because all the three algorithms are comparable. While LOF uses an index structure like R* trees to find nearest neighbours, this index structure collapses with high dimensional spaces and the time complexity of all three methods becomes in the order $O(N^2)$ when the dimensionality is above 15 or 20 dimensions. In those cases, outlier analysis can benefit from linear approaches approximating *k*-nearest neighbours such as [8, 9, 10].

7 Summary and Conclusion

Outlier mining provides unprecedented advantages for detecting inconsistent records from a large database, which could not be possibly accomplished with traditional statistical techniques. Nevertheless the current popular outlier mining algorithms, when applied for engineering applications are neither efficient nor effective because of their domain-dependent parameters. The outlier notions in the current algorithms are targeted at either global or local outliers, but in engineering data, the data is typically noisy and clusters in the dataset are not well bounded as in the synthetic datasets used by some authors. At the same time, the current outlier mining algorithms cannot efficiently detect outliers from a dataset containing clusters of arbitrary shapes, such as in the engineering data.

The resolution-based outlier definition and the nonparametric RB-outlier mining algorithm RB-MINE introduced in this paper are more suited to engineering data when compared with the popular DB-outlier and LOF-outlier schemes. The algorithm overcomes the problems stated above and can be used for robust outlier detection in a wide variety of multi-dimensional datasets in engineering data.

The ability of mining top-N outliers based on ROF provides a mechanism of ranking by the degree of "interestingness". Problems in the records can be identified by looking at the top listed outliers for further investigation. The outlier mining technique we propose is expected to eliminate the burden of domain experts spent on estimating and tuning unknown parameters.

References

[1] Raz O., Buchheit R., Shaw M., Koopman P. and Faloutsos C. (2004). "Detecting Semantic Anomalies in Truck Weigh-in-Motion Traffic Data Using Data Mining." *Journal of Computing in Civil Engineering*, ASCE. Vol. 18, No. 4. pp.291~300.

[2] Knorr E., and Ng R. (1998). "Algorithms for Mining Distance-based Outliers in Large Datasets." *Proc. of 24th International Conference on Very Large Databases*.

[3] Breunig M., Kriegel H., Ng R. and Sander J. (2000). "LOF: Identifying Density-Based Local Outliers." *Proc. of ACM SIGMOD 2000 International Conference on Management of Data*, Dallas, TX.

[4] Tang J., Chen Z., Fu A. and Cheung D. (2002). "Enhancing Effectiveness of outlier Detections for Low Density Patterns." *Proc. of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, Taipei, Taiwan. pp. 535 - 548

[5] Foss A. and Zaïane O. (2002). "A Parameterless Method for Efficiently Discovering Clusters of arbitrary Shape in Large Datasets." *Proc. of 2002 IEEE International Conference on Data Mining (ICDM'02)*, Maebashi City, Japan

[6] Hawkins D. (1980). "Identification of Outliers". Chapman and Hall, London. pp.1.

[7] Ramaswamy S., Rastogi R. and Shim K. (2000) "Efficient Algorithms for Mining Outliers from Large Data Sets." In Proc. Of the ACM SIGMOD International Conference on Management of Data, Dallas, TX.

[8] Goldstein J. and Ramakrishnan R. (2000) "Constrast Polots and P-Sphere Trees: Space vs. Time in Nearest Neighbor Searches." In Proc. 26th VLDB conference.

[9] Kushilevitz E., Ostrovsky R. and Rabani Y., (1998) "Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces", STOC'98.

[10] Liu T., Moore A.W., Gray A., and Wang K. (2004) "An Investigation of Practical Approximate Nearest Neighbor Algorithms", NIPS, December.